

Prácticas Fundamentos de Redes Chat

Rubén Morales Pérez
Francisco Javier Morales Pérez

4 de diciembre de 2016

- 1 Introducción
- 2 Servidor
- 3 Usuario
 - Representación
 - Conversaciones
 - Mejoras
 - Varias personas en un chat
- 4 Configuración
- 5 Características del chat
- 6 Funcionamiento
 - Servidor
 - Cliente
 - Escribir mensajes

Chat

En esta práctica crearemos un chat multiusuario usando el paradigma cliente-servidor. El servidor será un proceso alojado en alguno de los ordenadores de la comunicación.

Usaremos como host la dirección IP local.

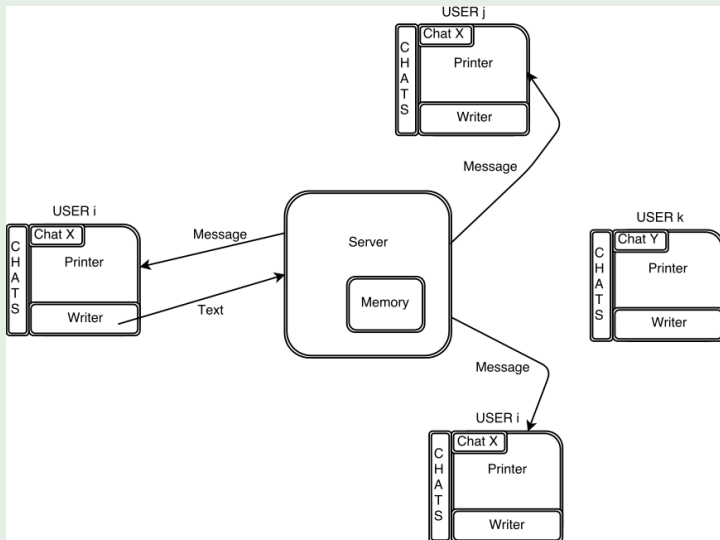
Cuidado

Hay redes wifi que bloquean ciertas comunicaciones, por eso hay que tener especial cuidado a la hora de ejecutar los programas.

Esquema

El esquema que usaremos será una clase *Writer* y una clase *Printer*, la primera se encargará de mandar los mensajes al servidor, la segunda de recibirlos y mostrarlos con el formato adecuado.

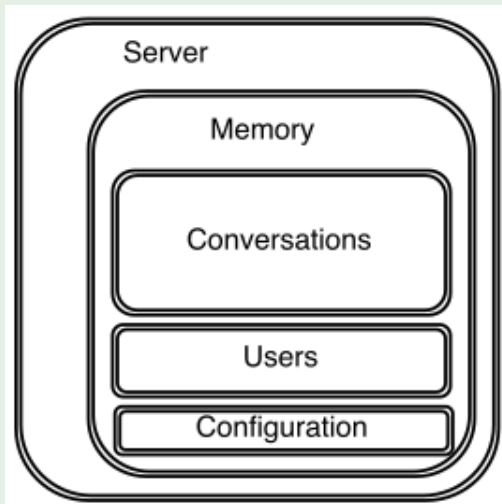
Esquema



Servidor

El servidor será una clase Server que estará continuamente recibiendo y mandando mensajes a los usuarios correspondientes. El servidor tiene una serie de archivos guardados para su organización.

Esquema



Representación

Un usuario quedará identificado por su idUser, los usuarios se guardarán en una carpeta especial con ficheros como el siguiente.

0.usr

```
0
ruben
password1
0
```

- Identificador de usuario
- Nombre de usuario
- Contraseña
- Grupos a los que pertenece

Mensajes

Los mensajes tienen un identificador del grupo al que pertenecen, del usuario que los envía y un identificador del mensaje. Estos tres parámetros formarán la clave primaria, también se incluirá la fecha del envío del mensaje y el nombre del usuario que lo envía.

Message

idConversation

idUser

idMessage

Text

Fichero conversación

Un fichero de conversación será una sucesión de mensajes.

example.chat

```
0 0 0 20/11/2016 at 06:42:01 — "Message 1"
0 1 1 20/11/2016 at 06:42:01 — "Message 2"
0 2 2 20/11/2016 at 06:42:01 — "Message 3"
0 3 3 20/11/2016 at 06:42:01 — "Message 4"
0 1 4 20/11/2016 at 06:42:01 — "Message 5"
0 3 5 20/11/2016 at 06:42:01 — "Message 6"
0 2 6 20/11/2016 at 06:42:01 — "Message 7"
0 1 7 20/11/2016 at 06:42:01 — "Message 8"
0 1 8 20/11/2016 at 06:42:01 — "Message 9"
0 4 9 20/11/2016 at 06:42:01 — "Message 10"
```

Varios chats del mismo usuario

Para tener varios chats podríamos usar:

- Varios Printer por usuario
- Un único Printer que guarda las conversaciones en archivos

Grupos

En este caso el cliente actuaría también como servidor que recibe los mensajes del servidor principal y los muestra por pantalla.

Configuración

El servidor alojará en ficheros las conversaciones, los usuarios y sus ajustes.

Clase Config

En la clase tendremos los directorios necesarios para guardar la información.

Habrà variables para controlar los usuarios y conversaciones tenemos registrados en el sistema, los puertos usados para las comunicaciones y el host del servidor.

Guardadas las extensiones de los ficheros de usuarios y conversaciones. Desde el servidor podremos acceder al fichero de cada conversación o usuario.

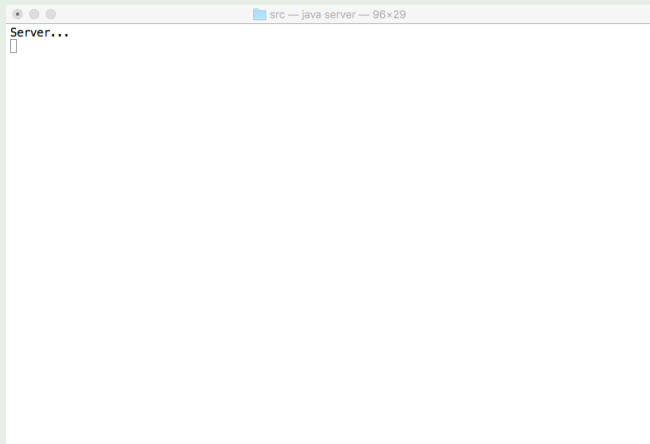
Debemos guardar la configuración al cerrar el servidor.

Posibles personalizaciones

- Negrita entre **
- Parpadeo entre \\
- Subrayado entre _ _

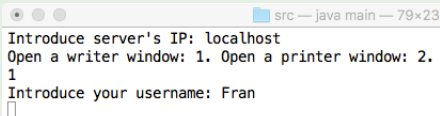
Servidor

La interfaz del servidor es sencilla.



Cada usuario tiene un terminal para el Writer y otro para el Printer.

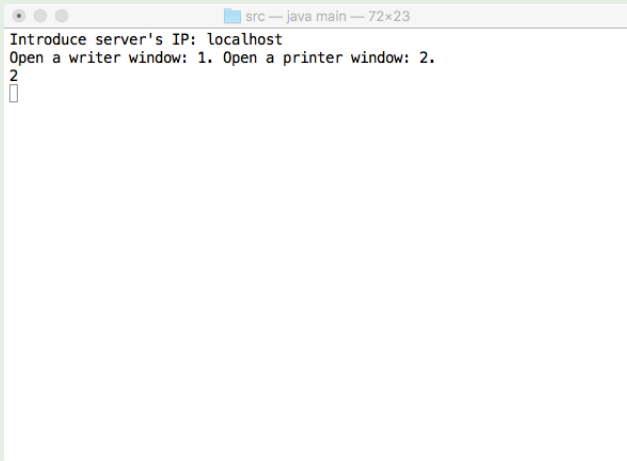
Fran writer



```
src — java main — 79x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Fran
█
```

Cada usuario tiene un terminal para el Writer y otro para el Printer.

Fran printer



```
src — java main — 72x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
2
█
```


Cada usuario tiene un terminal para el Writer y otro para el Printer.

Ruben writer

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Ruben
```

Cada usuario tiene un terminal para el Writer y otro para el Printer.

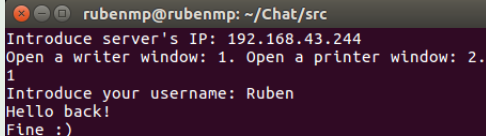
Ruben printer

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
2
```

Escribir mensajes

Ahora procedemos a escribir mensajes desde ambos ordenadores.

Ruben escribiendo

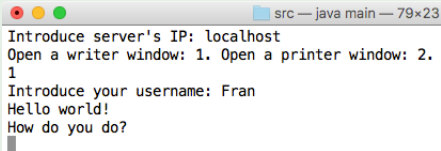
A terminal window with a dark purple background and a grey title bar. The title bar contains window control icons and the text 'rubenmp@rubenmp: ~/Chat/src'. The terminal text shows a sequence of commands and responses: 'Introduce server's IP: 192.168.43.244', 'Open a writer window: 1. Open a printer window: 2.', '1', 'Introduce your username: Ruben', 'Hello back!', and 'Fine :)'.

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Ruben
Hello back!
Fine :)
```

Escribir mensajes

Ahora procedemos a escribir mensajes desde ambos ordenadores.

Fran escribiendo

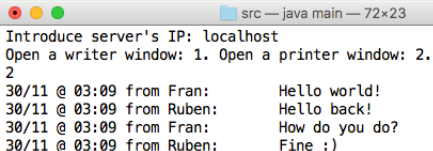


```
src — java main — 79x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Fran
Hello world!
How do you do?
█
```

Escribir mensajes

Comprobamos que funciona

Fran recibiendo



```
src — java main — 72x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
2
30/11 @ 03:09 from Fran:      Hello world!
30/11 @ 03:09 from Ruben:    Hello back!
30/11 @ 03:09 from Fran:    How do you do?
30/11 @ 03:09 from Ruben:    Fine :)
█
```

Escribir mensajes

Comprobamos que funciona

Ruben recibiendo

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
2
30/11 @ 03:09 from Fran:      Hello world!
30/11 @ 03:09 from Ruben:    Hello back!
30/11 @ 03:09 from Fran:    How do you do?
30/11 @ 03:09 from Ruben:    Fine :)
```